REMARKS

In the Office Action mailed December 9, 2003, the drawings and the specification were objected to. Claims 1, 4-6, and 8-12 were rejected under 35 U.S.C. §112, second paragraph as being generally narrative and indefinite, failing to conform to current U.S. practice; claims 1, 3, 4, 7, and 9-12 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Donohue (U.S. Patent No. 6,199,204) further in view of Davis et al. (U.S. Patent No. 6,282,712); claim 2 was rejected under 35 U.S.C. 103 as being unpatentable over Donohue (U.S. Patent No. 6,199,204) in view of Davis et al. (U.S. Patent No. 6,282,712) and further in view of Chan et al. (U.S. Patent No. 5,276,881); claims 5-6 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Donohue, further in view of Davis et al. and further in view of Hocker et al. (U.S. Patent No. 5,943,678); and claim 8 was rejected under 35 U.S.C. § 103(a) as being unpatentable over Donohue, further in view of Davis et al. and further in view of Reisman (US2002/0124055 A1).

The foregoing objections and rejections are respectfully traversed.

In accordance with the foregoing, the drawings, specification, and claims 1-12 have been amended. A substitute specification is included herewith. Claims 1-12 are pending and under consideration. No new matter is presented.

References –

Donohue discusses an updater component which "...may be delivered to customers with [data] fields set to null values, and then the installation procedure includes an initial step of the updater component interrogating its software product to obtain an identifier and current program version and release number...the software vendor may pre-code the relevant product ID and version number into the updater component." (Col. 7, lines 46-54). Donohue also discusses obtaining the version of application that is installed in a computer each time a predetermined time counted by a timer passes.

Further, in Donohue, it is determined whether updating of the application is necessary or not by comparing the version of the application with the contents of an update list made public on the network, and performing a predetermined process. (Col. 11, lines 26-67). Donohue further discusses "...information for identifying one or more locations is held by said updater component and includes a product identifier of a computer program product, the updater component being adapted to provide said product identifier to a search engine, the product

identifier serving as a search parameter for use by said search engine to identify network locations." (Cols. 18-19, claim 3).

In Donohue, a product identifier is used to determine whether a program needs to be updated.

Davis et al. discuss the automatic installation of "software in a heterogeneous environment". (Col. 3, lines 19-20). Further, Davis et al. discuss providing a "...hardware and software inventory support, the centralized management system provides a listing of the hardware and software components on the computers in the distributed system...software update functionality performed by the centralized management system installs new software versions when the current version of the software becomes outdated." (Col. 3, lines 47-57). Davis et al. also discuss automatic installation of "the appropriate services...according to the preferences of the administrator". (See Col. 11, lines 56-67 and Col. 12, lines 1-8).

In Davis et al., software updates are performed according to the preferences of the administrator.

Chan et al. discuss "...computing environments which use deviations, modifications, and extensions of the ANSI and OSF standards." (Col. 8, lines 59-61). Further, Chan et al. discuss replacing library function names "...by unique keywords using function-like macro definitions in the modified ANDF version of the header file..." (Col. 46, lines 64-67).

In Chan et al., library function names are updated by function-like macro definitions.

Hocker et al. discuss providing "...an improved system and method for retrieving prior versions of an application, database, or other function." (Col. 1, lines 54-57). Further, Hocker et al. discuss that "...the user is allowed to examine older versions of the data without confusing older data with present data." (Col. 2, lines 51-53). Hocker et al. discuss the use of "...geometrical calculations known to those skilled in the art, that the source icon was dragged to the VTT icon, then a list of prior versions, sorted for example by date, of the application, database, or other information...represented by the source icon is displayed". (Col. 2, lines 62-67).

In Hocker et al., a list of prior version of applications can be obtained by the use of geometrical calculations.

Reisman discusses providing software that "...performs a plurality of functions including a fetch function for fetching a first data object from one of the data sources, and a pre-fetch function for automatically pre-fetching a plurality of additional data objects referenced by the

first data object from respective other ones of the independently-operated data sources identified by information embedded in the first data object". (Page 3, paragraph 0025). Reisman also discusses updating software by user-entered or vendor-entered product identification to schedule updates periodically. (Page 4, paragraphs 0047-0050). Further, Reisman discusses a communications module that "...verifies that all send objects are as specified, that all fetch objects are scheduled to be available, verifies that sufficient disk space is available for all fetch objects and for compressed transmission copies of all objects..." (Page 7, paragraph 0089).

In Reisman, software can be updated by user-entered or vendor-entered product identification that allows periodically scheduled updates.

Donohue in view of Davis et al. discusses the automatic updating of software based on the preferences of an administrator.

Donohue in view of Davis et al. and in further view of Chan et al. discusses replacing library function names with the help of function-like macro definitions.

Donohue in view of Davis et al. and in further view of Hocker et al. discusses the use of "geometrical calculations" to aid in retrieving prior versions of an application, database, or other function.

Donohue in view of Davis et al. and in further view of Reisman et al. discusses updating software by user-entered or vendor-entered product identification to schedule updates periodically.

Rejection under 35 U.S.C. §112, second paragraph rejection –

According to the Examiner's suggestions, we have amended claims 1, 4-6, and 8-12 to overcome the rejections of same under 35 U.S.C. §112. Please note that we have also amended claims 2, 3, and 7, in addition to claims 1, 4-6, and 8-12, for clarification.

Motivation to Combine References –

The Applicant requests the Examiner to kindly consider that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either explicitly or implicitly in the references themselves or in the knowledge generally available to

one of ordinary skill in the art.

The Applicant respectfully disagrees with the Examiner and requests to withdraw all rejections based on the combination of references used to reject all pending claims 1-12 under 35 U.S.C. § 103(a).

"The test for an implicit showing is what the combined teachings, knowledge of one of ordinary skill in the art, and the nature of the problem to be solved as a whole would have suggested to those of ordinary skill in the art." In re Kotzab, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000). See also In re Lee, 277 F.3d 1338, 1342-44, 61 USPQ2d 1430, 1433-34 (Fed. Cir. 2002) (discussing the importance of relying on objective evidence and making specific factual findings with respect to the motivation to combine references); In re Fine, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988); In re Jones, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

If it is assumed that there is sufficient motivation to combine the references, then the present invention is still not obvious over the cited references either standing alone or in combination.

Rejection under 35 U.S.C. § 103(a) – Combined References

In the Office Action, the Examiner asserts that the "feature point" corresponds to "product identifier" in Donohue and "version of application" corresponds to "release number". (Col. 8, lines 31-36).

The Examiner's assertions are respectfully traversed.

In contrast to the foregoing references relied upon, in the present invention, the "feature point" as recited in independent Claim 1 or the "at least one of a plurality of feature points" as recited in the remaining independent Claims 4 and 9-12, is a format designation or a macro instruction stored in the data file. The "product identifier" in Donohue is defined in Col. 4, lines 37-45:

> "The information for use in identifying a network location may be explicit network location information or it may be a software vendor name or any other information which can be used as a search parameter for identifying the location. In the preferred embodiment, the information is a product identifier which is provided by the updater component to a search engine to initiate a search to identify the relevant network location at which are stored the software resources for implementing updates to that product."

The "product identifier" is not a format designation or a macro instruction as in the

present invention. Therefore, the recitation of a "feature point" or "a plurality of feature points" in all independent claims of the present invention, is not the same as the "product identifier" in Donohue. Further, the application program includes a list relating to feature points usable in the version of the application program.

In contrast, in the present invention, in Claim 1 and all other independent claims including Claims 4 and 9-12, the "feature point" of the data file is extracted when some data file is used, to judge whether the data file can be used by the installed application program. As an example, independent Claim 1 of the present application recites in relevant part:

> "...extracting a feature point after having analyzed a data file,
> selecting a version of the application program suited for the data file based on the extracted feature point..."

Further, according to the present invention, all independent claims including Claims 1, 4, and 9-12, recite that in selecting a version of the application program suited for the data file based on the extracted feature point, it is not necessary to consider when the application program should be updated. However, in Donohue the time of updating computer programs is discussed. (Col. 11, lines 50-53).

Also, a user can update the application program only if the application program should be updated when the data file cannot be used by the present version of the application program. Accordingly, user can update the application program if necessary.

Donohue fails to disclose or suggest any such feature disclosed in the present invention.

Donohue in view Davis et al. do not discuss extracting a "feature point", which is a format designation or a macro instruction, as in the present invention. The feature point, as recited in all independent claims – 1, 4, and 9-12, is extracted from a data file in order to decide which version of the application program is to be selected and installed. As an example, independent Claim 4 of the present application recites in relevant part:

> "...extracting at least one of a plurality of feature points after having analyzed at least two or more data files,
> selecting a version of the application program in which a data file is readable based on the feature point..."

In contrast, Davis et al. discuss the automatic updating of software based on the preferences of an administrator.

The Applicant, therefore, respectfully submits that there is no suggestion or motivation

to combine Davis et al. and Donohue, by one skilled in the art. Therefore, the Applicant requests the Examiner to traverse this rejection.

Donohue in view of Davis et al. and in further view of Chan et al. discuss replacing library function names with the help of function-like macro definitions. Chan et al. do not discuss extracting a "feature point", as in the present invention, from a data file in order to decide which version of the application program is to be selected and installed.

Moreover, in the present invention, as recited in claims 1-12, a feature point, which is a format designation or a macro instruction, is used to identify which version of an application program should be installed. As an example, independent Claim 9 of the present application recites in relevant part:

> "...extracting at least one a plurality of feature points after having analyzed at least two or more data files,
> selecting a version of the application program which can read the readable data file based on the feature point..."

Chan et al., on the other hand, do not discuss the use of a format designation or a macro instruction to install application programs.

The Applicant, therefore, respectfully submits that there is no suggestion or motivation to combine Chan et al. with Davis et al. and Donohue, by one skilled in the art. Therefore, the Applicant requests the Examiner to traverse this rejection.

Hocker et al. do not discuss extracting a "feature point", as in the present invention, from a data file in order to decide which version of the application program is to be selected and installed. Further, Donohue in view of Davis et al. and in further view of Hocker et al. discusses the use of "geometrical calculations" to aid in retrieving prior versions of an application, database, or other function.

Moreover, in the present invention, as recited in claims 1-12, a "feature point", which is a format designation or a macro instruction, is used to identify the appropriate version of an application program to be installed. As an example, independent Claims 10 and 11 of the present invention recite in relevant part:

> "...an extracting unit extracting at least one of a plurality of feature points...
> a selecting unit selecting a version of the application program ..."

However, Hocker et al., on the other hand, discuss the use of geometrical calculations to obtain a list of prior versions of applications etc.

The Applicant, therefore, respectfully submits that there is no suggestion or motivation to combine Hocker et al. with Davis et al. and Donohue, by one skilled in the art. Therefore, the Applicant requests the Examiner to traverse this rejection.

Reisman does not discuss extracting a "feature point", as in the present invention, from a data file in order to decide which version of the application program is to be selected and installed. Further, Donohue in view of Davis et al. and in further view of Reisman et al. discusses updating software by user-entered or vendor-entered product identification to schedule updates periodically.

Moreover, in the present invention, as recited in claims 1-12, a "feature point", which is a format designation or a macro instruction, is used to determine the appropriate version of an application program to be installed. As an example, independent Claim 12 of the present application recites in relevant part:

> "...extracting at least one of a plurality of feature points after having analyzed a data file,
> selecting a version of the application program suited for the analyzed data file based on an extracted feature point..."

However, Reisman, on the other hand discusses updating software by user-entered or vendor-entered product identification to schedule updates periodically.

The Applicant, therefore, respectfully submits that there is no suggestion or motivation to combine Reisman with Davis et al. and Donohue, by one skilled in the art. Therefore, the Applicant requests the Examiner to traverse this rejection.

Concluding Remarks –

Thus, the present invention is not obvious over Donohue or any combination of any other references thereof because Donohue fails to disclose or suggest the above features of the present invention.

Withdrawal of the foregoing rejections is respectfully requested.

There being no further outstanding objections or rejections, it is submitted that the application is in condition for allowance. An early action to that effect is courteously solicited.

Finally, if there are any formal matters remaining after this response, the Examiner is requested to telephone the undersigned to attend to these matters.

If there are any additional fees associated with filing of this Amendment, please charge the same to our Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

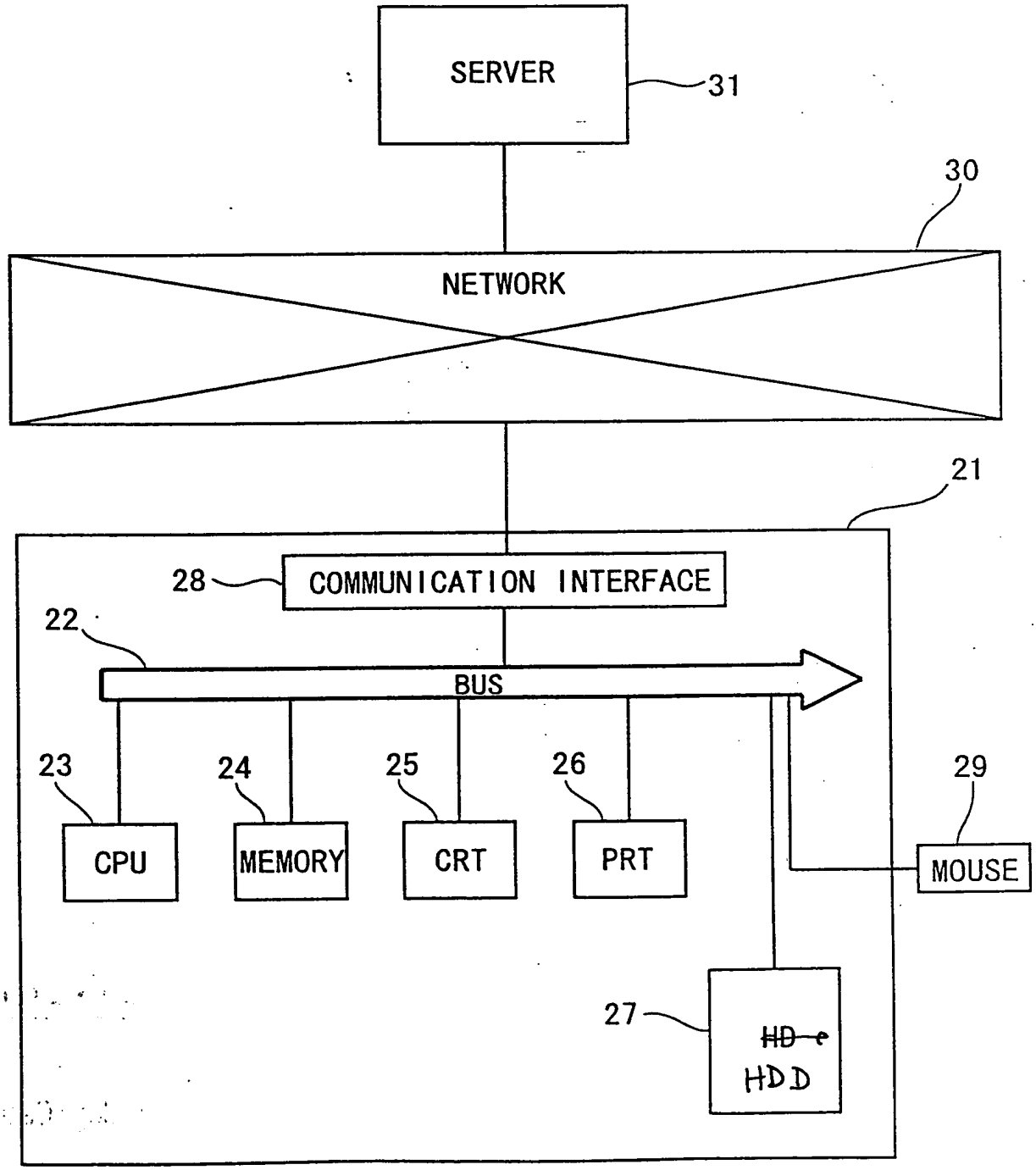
Date: April 9, 2004

By: 
Gene M. Garner II
Registration No. 34,172

1201 New York Avenue, NW, Suite 700
Washington, D.C. 20005
Telephone: (202) 434-1500
Facsimile: (202) 434-1501

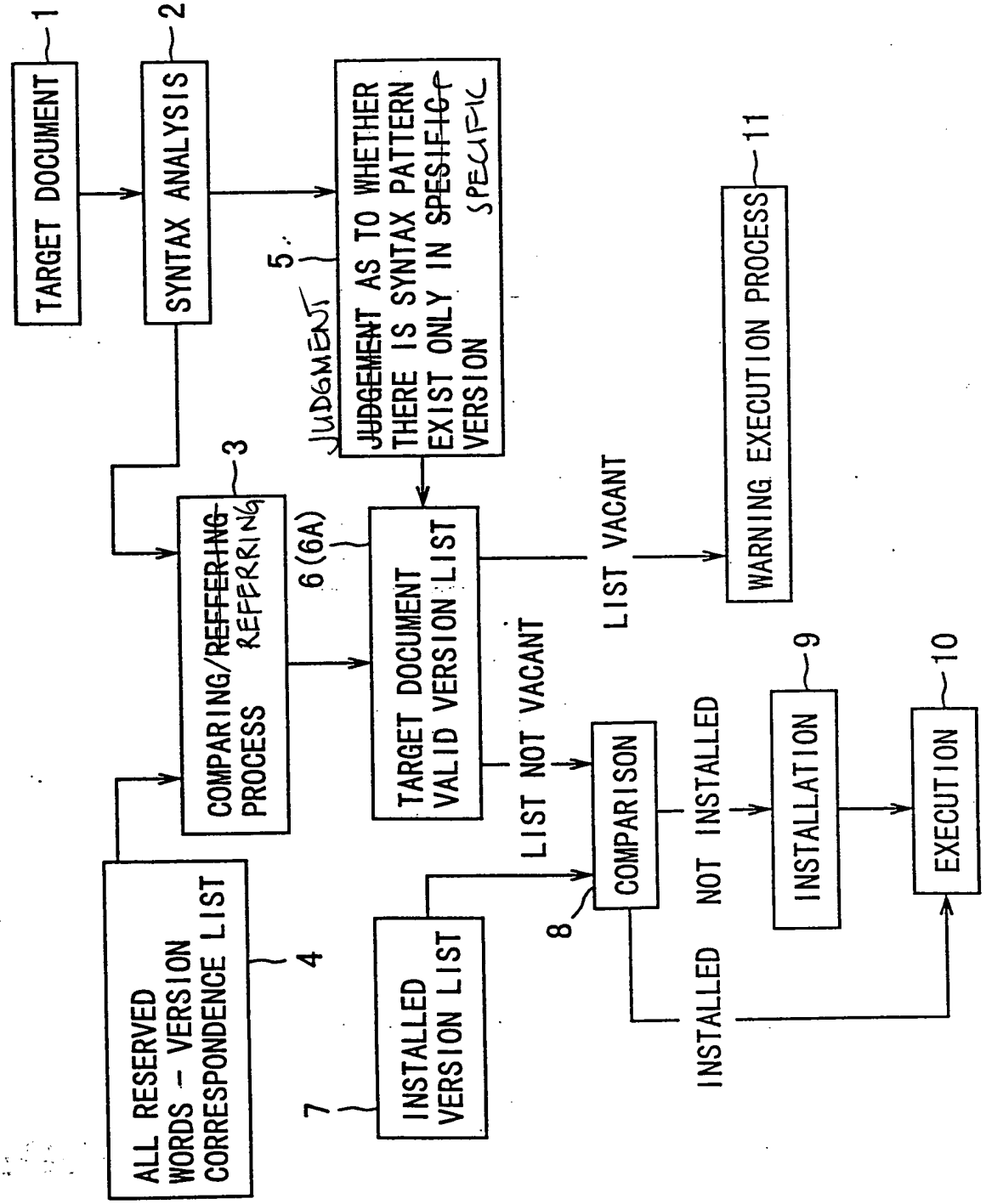
ANNOTATED SHEET

FIG. 2



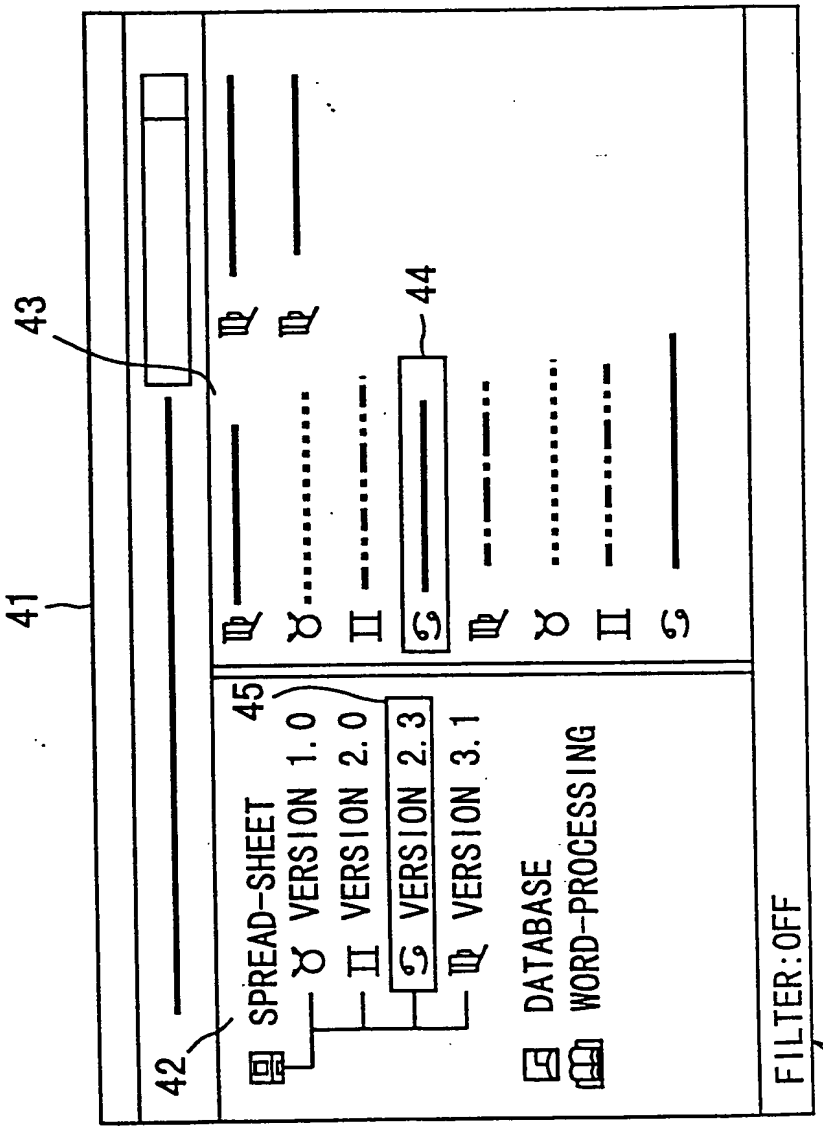
ANNOTATED SHEET

FIG. 1



ANNOTATED SHEET

FIG. 6



USUALLY, THERE ARE MANY CASES THAT THE DOCUMENT WHICH IS MADE BY THE SAME APPLICATION ~~ACCOMPANYS~~ ACCOMPANIES THE SAME ICON, REGARDLESS OF ITS VERSION. HOWEVER, FOR EXAMPLE, AS SHOWN IN THE DRAWING, IF THE DOCUMENT ICON IS DISPLAYED WITH BEING REPLACED INTENTIONALLY ACCORDING TO A VERSION, A CONVENIENCE IS GIVEN TO USERS.

Docket No.: 1046.1234

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

Takahiro MASUDA, et al.

Serial No. 09/764,352 Group Art Unit: 2122

Confirmation No. 7139

Filed: January 19, 2001 Examiner: Chow, Chih-Ching

For: INSTALLATION METHOD, ACTIVATION METHOD, EXECUTION APPARATUS AND MEDIUM OF APPLICATION PROGRAM

AMENDED SPECIFICATION

INSTALLATION METHOD, ACTIVATION METHOD, EXECUTION APPARATUS AND MEDIUM OF APPLICATION PROGRAM

BACKGROUND OF THE INVENTION

1. Field of the Invention

[0001] ~~This~~The present invention relates to the technology of automatic installation and automatic activation of an application of a version necessary for data.

2. Description of the Related Art

[0002] Data form of a document file made with a personal computer and a word processor, and data form of a spreadsheet file is changed according to version up of an application program which is used to make/update the data. Therefore, there are many cases that data made with a program of a new version ~~can not~~cannot be used by a program of an old version. On the contrary, there is the case that the data which was made with an application of an old version ~~can not~~cannot be edited by an application of a new version.

[0003] In case of application of a word processor, data storing form is different due to a version of an application. As a result, there is the case that layout of a document cannot be reproduced with an application program of another version. Therefore it is necessary to convert data form using another application called a document converter. In addition, the usable object is different due to its version in case of spreadsheet program. Therefore, there are cases ~~that~~in which errors occur when reading data ~~made~~ with a different version ~~with~~of a one program.

[0004] Furthermore, the facility program that a constant procedure called macro is described is attached in this kind of program. This macro-specification, presence of

reserved word, for example, ~~are~~is different due to a version of the program. Therefore, macro concerned may have produced execution impossibility or an error when opening data file with a program of a version different from the version with which the data was made.

[0005] In networked society represented with Internet, there are many cases ~~that~~in which this kind of data file is exchanged via communication. Thus, it was a big problem whether an application program installed in ~~a~~one's computer ~~of oneself~~ could be applicable for the version of the received data.

[0006] Therefore, a procedure ~~below~~ was necessary in case that treating the data file with a version of the application installed in the user's computer was impossible. The user, judges a version of the application that can be used, ~~investigate~~investigates an acquisition point of the application, and ~~install~~installs it ~~in the~~on one's computer, ~~of oneself~~ after the user obtained an application.

[0007] ~~By the way, in~~In case ~~of~~a data file ~~which~~is given a different extension ~~is given~~ due to a version, the applicable version of the application program can be known by just referring to extension.

[0008] However, like a document form of "doc", there are cases ~~that~~when the universal extension is applied even if versions of the application program with which the document was made are different. In this case the judgment of an appropriate version was difficult in just observing data file from the outside.

[0009] Therefore, in the worst case scenario, repeating the following procedure was necessary. A user should try to install an application program of various kinds of

versions with cut and try and repeats opening the data file.

[0010] The present invention has been made to solve the above-explained problems, and therefore, has an object to provide such a technique capable of automatically installing an application program of a version optimally selected for a data file, thus eliminating a cumbersome version judging operation by a user, and automatically installing application program with the optimum version.

SUMMARY OF THE INVENTION

[0011] Firstly, this invention analyzes the data file which is to be read, and extracts a characteristic point. ~~And a~~A version of an application suited for the data file is distinguished by such an extracted characteristic point. Secondly, it is ~~judged~~determined whether an application program of the version distinguished is already installed ~~already~~. An application program of a version concerned is newly installed ~~newly~~ when found not installed by this judgment result.

[0012] For example, this characteristic point ~~is~~may be a reserved word and/or a syntax pattern of macro instruction in a document file.

[0013] In many cases, these reserved words and the syntax pattern depend on a version of the application program. A version of the most suitable application program to open a document file becomes clear by extracting these reserved words and/or syntax patterns. Based upon the result of an analysis ~~result~~ of various versions of application programs, both a document file and application programs of various corresponding versions are displayed at the same time on a display apparatus.

[0014] Then, users may select any one of these application programs to open this document file among various versions.

[0015] In ~~this case~~the present invention, based on information provided by an analysis result, the document file may be displayed with the icons which differ in each version. Alternatively, when a user first selects a certain application program, only document files ~~which~~that can be opened by this selected application program may be displayed.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is ~~a function block figure of~~ an automatic installation system according to an ~~Embodiment~~embodiment of this invention.

Fig. 2 is ~~a figure of block showing~~ the system constitution of Embodiment 1.

Fig. 3 is ~~a figure of block showing~~the relationship of each list of Embodiment 1.

Fig. 4 is a screen display example in visual display unit of Embodiment 2.

Fig. 5 is a ~~function block figure to realize~~ screen constitution of figure 4.

Fig. 6 is a screen display example in visual display unit of Embodiment 2 ~~(2)~~.

Fig. 7 is a screen display example in visual display unit of Embodiment 2 ~~(3)~~.

Fig. 8 is ~~a function block figure which shows~~ a document file's relationship with an application program in Embodiment 2.

Fig. 9 is a figure ~~of function block~~showing when an application program of a corresponding version does not exist in an application list as a result of analysis according to a document analysis control unit in Embodiment 2.

Fig. 10 is a ~~function block figure in case~~ method of installing an application program based on a document analysis result in Embodiment 2.

Fig. 11 is a ~~function block figure in case~~ method of uninstalling when an installation domain is insufficient in Embodiment 2.

Fig. 12 is a ~~function block figure which explains a~~ procedure to execute uninstallation and get a domain in Embodiment 2.

Fig. 13 is a ~~figure of~~ sequence of steps showing the method of ~~at~~ downloading an application program through a network in Embodiment 2 ~~(1)~~.

Fig. 14 is a ~~figure of~~ sequence of steps showing the method of ~~at~~ downloading an application program through a network in Embodiment 2 ~~(2)~~.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

~~[~~Embodiment 1~~]~~

[0016] ~~Based on a drawing, embodiments of this invention are described as follows.~~ Fig. 1 is ~~a figure of block that function block figure of~~ an automatic installation system according to an ~~Embodiment~~embodiment of this invention, and Fig. 2 shows system constitution of this ~~Embodiment~~embodiment.

[0017] Terminal 21 is connected to server 31 through network 30 as shown in Fig. 2. ~~A terminal~~Terminal 21 ~~itself~~ constitutes a computer system and can carry out automatic installation processing of this ~~Embodiment~~embodiment ~~only with terminal 21~~.

[0018] Terminal 21 has a Central Processing Unit (CPU) 23, a memory 24, a visual display unit (CRT) 25, a print unit (PRT) 26 and Hard Disk Drive (HDD) 27 through bus 22. In addition, terminal 21 has a mouse 29 as an external auxiliary input means.

In addition, the bus 22 is connected to the network 30 through communication interface 28.

[0019] Fig. 1 shows an automatic installation procedure of this ~~Embodiment~~embodiment. A program stored in HDD 27 is read into the memory 24, and the CPU 23 executes the program sequentially. Installation is realized in such a procedure.

[0020] The target document 1, as shown in Figure 1, is, for example, data file for a document file and a spreadsheet for a word processor. The target document 1 is registered in HDD 27. The target document 1 has a normal document domain and a macro-description domain. Processing and a correction of character string prescribed with the macro instruction that accompanied the document become possible when reading the target document 1 with a predetermined word processor program.

[0021] The CPU 23 executes syntax analysis process 2 for this target document 1 after such target document 1 is read into the memory 24. This syntax analysis process is, specifically, an operation to extract macro instruction reserved word.

[0022] Extraction of macro instruction reserved word is performed by generating the reserved word list 12 from the target document 1. This extraction is performed as follows~~, specifically~~. All the reserved words are referred to a database (DB) 13 registered with HDD 27. The reserved word from the inside of the target document 1 is extracted by pattern matching. The extracted reserved word is registered in reserved word list 12. The list 4 registered in the HDD 27 is compared with this reserved word list 12 (comparing/referring process 3). At last, a valid version list 6 for the target document 1 is generated.

[0023] In addition, CPU ~~(CPU)~~ 23 executes search of a syntax pattern in parallel

with above-mentioned syntax analysis process 2 (step 5). A macro-program in target document 1 is divided to a token, an attribute of a token is judged, and it is judged whether a combination state of a token of a specific attribute exists. The detection of a syntax pattern is realized in this way. The detection of a coupling state implies, for instance, detection of a coupling state "variable which constitutes instance of Collection-Class" + "(" + "variable which constitutes instance of Strings-Class" + ") ".

[0024] When a valid version list 6a is generated, a detection result of this syntax pattern is reflected and a final valid version list ~~14~~6a is generated. When the version list 6a is generated, either an extraction result of macro instruction reserved word or a detection result of a syntax pattern may be given priority to. For example, in the case ~~that~~ the version produced based upon the extracted result of the macro instruction reserved word is different from the version produced based upon the detection result of the syntax pattern~~, the~~. The latest version having the larger numeral value may be employed with a priority so as to produce the valid version list 6a.

[0025] Secondly, an installed version list 7 is compared with the valid version list 6a (comparison process 8). When an application program of a corresponding version is installed, in the already executable state in HDD 27, in terminal 21 as a result of this comparison process 8, the application program is executed (execution process 10). When an application program of a corresponding version is not installed, installation process 9 is performed.

[0026] It should also be noted that when the valid version list 6 (6a) is produced, if the content of the version list is empty, namely if the valid version cannot be judged, then the ~~central processing unit (~~CPU~~)~~ 23 executes a warning process 11. For

example, as for this warning process 11, a visual warning display may be performed on the CRT 25.

{Embodiment 2}

[0027] In Embodiment 1, the method for automatically determining the optimum version of the relevant application program in such a manner that the CPU 23 executes the comparing process operation 8 is described. This Embodiment 2 has the following characteristics. A result of comparison process 8 is displayed in the form that at a glance a user can grasp on the CRT 25. ~~And the room where~~The user may then ~~oneself selects~~select an application program ~~of~~having more than one version ~~exists~~.

[0028] It should be understood that since the process operations defined prior to the comparing process operation 8 are similar to those of Embodiment 1 among the process operations explained in Fig. 1, explanations thereof are omitted.

[0029] While the ~~central processing unit (~~CPU~~)~~ 23 compares the valid version list 6(6a) obtained from the target document 1 with the installed version list 7, this ~~central processing unit (~~CPU~~)~~ 23 displays a comparison screen 41 as indicated in Fig. 4 on the CRT 25. This comparison screen 41 is divided in application list display window 42 of a left side of the ~~drawing~~figure, and module list display window 43. In application list display window 42, an application program can be displayed in a hierarchical manner for each group. In the ~~drawing~~figure, version 1.0-3.1 are listed as an application program in a folder of a spreadsheet (a spreadsheet program).

[0030] In addition, the database program and the word processor program are in a closed state, but have a plurality of application programs in their folders as well.

[0031] On the other hand, in module list display window 43, more than one target document 44 (Fig. 1, target document 1) is displayed as a module. A user can decide optionally which document (a module) should be opened by which version of an application program by referring to CRT.

[0032] In ~~the drawing~~Fig. 4, for example, when the target document 44 (a module) is opened by an application program of version 2.0 of a spreadsheet, the mouse cursor 46 on a screen is arranged by the mouse 29, and a button of the mouse 29 (not illustrated) is pushed once here. By this act (drag), the target document 44 (a module) is selected, and only an application program of a version for opening this target document 44 is displayed with high brightness in application list display window 42. That is, ~~all of~~ a plurality of corresponding versions are displayed with high brightness when there are plurality of versions of application programs to open the target document 44.

[0033] An activation method of an application program will now be discussed. ~~is shown below.~~ The Mouse cursor 46 is moved on an application program of an arbitrary version (this place, version 2.0) on application display window 42 in the state a button of the mouse 29 being pushed (a drag state). When the button is opened up (drop), an application program of version 2.0 of a spreadsheet is activated as a target file with the target document 44 (a module). This technique is an input technique on screen interface so-called "drag & drop."

[0034] ~~The drawing where~~Fig. 4 and Fig. 5 describe function constitution to realize interface. ~~described in Fig. 4 is shown in Fig. 5. In the drawing, the~~The document analysis control unit 51 has a function to execute the syntax analysis process 2 and the

comparing/referring process 3, as shown in Fig. 1~~, and~~. The document analysis control unit 51, also has a function to control display on the CRT 25. That is, the document analysis control unit 51 sequentially reads the target documents 1 under a predetermined folder or a directory~~,~~ and executes analysis of the version and displays the target documents 1 in a list form on the module list display window 43.

[0035] In a module displayed on this module list display window 43 (target document 44), the following matters are distinguished ~~already~~ by the document analysis control unit 51: whether~~. Whether~~ the document file is a legitimate document file readable by an application program installed in HDD 27 or not, and with~~. With~~ which version the document can be opened when the document is a legitimate document file.

[0036] When document analysis control unit 51 reads the target document 1 and executes analysis, it notifies application path acquisition unit 54 of an application name and the version name which are the result of this analysis. This application path acquisition unit 54 controls a display of application list display window 42. ~~And this~~The application path acquisition unit 54 lets application list display window 42 display only an application of a version received from document analysis control unit 51.

[0037] When target document 44 on the module list display window 43 (a module) is selected by ~~said~~ drag operation of mouse 29, this choice information is notified to the application path acquisition unit 54 through the document analysis control unit 51. ~~And a~~A version of a corresponding application program on the application list display window 42 is displayed with high brightness.

[0038] When an application of a predetermined version (45 to program an

application of a spreadsheet of version 2.0 here) is selected by drop operation of mouse 29, this choice information is notified to ~~OS (~~Operating System~~)~~ (OS) as an execution pass through the application path acquisition unit 54. ~~By this, the~~The OS, thus, receives an execution file path from application path acquisition unit 54 and also receives a document file path from document analysis control unit 51. ~~And the~~The OS executes target document 44 (a module) by an application program of a selected version.

[0039] In addition, Windows 3.1, Windows 95 or Windows 98 of Microsoft Inc. ~~company~~ and other OS may be used as OS in this ~~Embodiment~~embodiment.

[0040] Fig. 6 and Fig. 7 are transformation examples of the comparison display in which a version of an application program corresponding to a target document (a module) and a target document ~~was~~are compared in a display of comparison screen 41.

[0041] Fig. 6 is the ~~drawing which is displayed~~figure with an icon of a corresponding version when a different icon is defined by each version of an application program and the target document 44 (a module) is displayed. Here, in the existing OS, an icon is defined by extension of a file name (for example, XLS and DOC).

[0042] Therefore, the file is displayed with the same icon even if a version of the file is different when the same extension is given to a file. ~~In this way, grasp~~Getting a grasp of a version of the target document ~~was~~used to be difficult when merely the icon was displayed.

[0043] Here, in an example shown by Fig. 6, the corresponding version is grasped

~~about~~by all target documents 44 in the document analysis control unit 51. Therefore, a version corresponding to a target document is displayed without being controlled by the presence of extension of a file name but being distinguished at first sight with an icon.

[0044] Fig. 7 is the example ~~that~~where only the target document 44 corresponding to a version appointed with the application list display window 42 is displayed in the module list display window 43. Here, when a specific version is appointed by the application list 53, the version is detected by the application path acquisition unit 54, and it is informed that the version is appointed towards the document analysis control unit 51.Based on this notice, only a target document corresponding to the version (module 44) is displayed on the module list display window 43 by the document analysis control unit 51.

[0045] Fig. 8 shows ~~the general idea that~~ a version of the application program 53 ~~is~~being selected by a target document. ~~By the~~The screen interface described in Fig. 4, Fig. 6, and Fig. 7, ~~the situation when~~aids the target ~~document~~documents 84a and 84b ~~are~~to be read in the document analysis control unit 51 ~~is~~, as described below. When these target documents are analyzed by the detailed analysis unit 51a, the application program 45a of the most suitable version is chosen among already installed application programs (Fig. 8~~, drawing~~). ~~And the~~The target document 84a is opened by the application program 45a decided by the detailed analysis unit 51a.

[0046] ~~A function block constitution when~~In function constitution, an application program of a version corresponding target document 44 does not exist in application list 53, ~~is~~ shown in Fig. 9, ~~as a result~~ that target document 44 is analyzed by document analysis control unit 51. ~~Because~~Since, most of the function constitution shown in Fig.

9 is similar to Fig. 5, a detailed explanation ~~of the part~~ is omitted ~~in the drawing~~.

[0047] The installation control unit 91, shown in Fig. 9, is the control unit for installing an application program in the state that execution is possible in HDD 27. Specifically, the installation control unit 91 has a function to install an application program in HDD 27 of the terminal 21 from a CD-ROM and server 31 that are not illustrated. When the application path acquisition unit 54 cannot acquire a path from the application list 53, in other words, when a version corresponding to the target document 44 selected in module list 52 (a module) does not exist, ~~how~~the method in which the installation control unit 91 is activated is shown in Fig. 9.

[0048] Fig. 10 is ~~a drawing where~~ an installation procedure of an application program of a new version by the installation control unit 91 is shown. In the ~~drawing~~figure, the target document 1001, 1002 of version n are read into the document analysis control unit 51 sequentially, and the detailed analysis unit 51a judges a version of an application program.

[0049] In spite of the judgment, an already installed application program (1003) is version m. It does not correspond to the target documents 1001 and 1002 of the version n (Fig. 10 (a)). A notice from the document analysis control unit 51 is received then, the installation control unit 91 is activated, and the application program 1004 corresponding to the version n of the target documents 1001,1002 is newly installed (Fig. 10 (b)).

[0050] Here, an installation source of the application program may be HDD 27 in the terminal 21, it may be the CD-ROM drive unit which is not illustrated, and from server

31 through network 30. When an application program is downloaded through network 30 by server 31, FTP (File Transfer Protocol) using TCP/IP may be used.

[0051] The application program 1004 newly installed in HDD 27 is activated, and the target document 1001 or the target document 1002 of a version corresponding to this application program 104 (Fig. 10 (c)) is opened.

[0052] When an application program of a necessary version is installed, ~~a function block drawing and~~ the ~~process~~ procedure for an installation domain to be kept are shown in Fig. 11 and Fig. 12.

[0053] In Fig. 11, the uninstallation control unit 1101 is added to the function constitution shown in Fig. 9. That is, even though the installation control unit 91 is activated by an example described in Fig. 9 and Fig. 10, there is the case that the domain where an application program of a new version is installed in HDD ~~(HD)~~ 27 is insufficient. In this case, the uninstallation control unit 1101 shown in Fig. 11 is activated. This uninstallation control unit 1101 is also realized as a program, and the uninstallation control unit 1101 is realized by the program being executed by the CPU 23.

[0054] Fig. 12 shows a ~~process~~ procedure of the uninstallation and installation. When the most suitable version is not detected by an application program installed by the detailed analysis unit 51a (Fig. 12 (a)), the installation control unit 91 starts.

However, there is the case that the domain where a new application program is installed, does not remain in HDD 27 (Fig. 12 (b)). In this case the installation control unit 91 notifies the uninstallation control unit 1101 of domain insufficiency. By this, the

uninstallation control unit 1101 eliminates an unnecessary file from HDD 27. It is decided which file is eliminated with a standard as follows.

[0055] For example, the application program that does not satisfy a utilization condition may be uninstalled with precedence. For example, an application program of a condition that it is usable after its download for 30 days corresponds to this. In addition, it can be uninstalled in volume order.

[0056] Generally, when importance of an application with a little volume is low, the volume is eliminated first from a small application program. In addition, the volume is eliminated from a big application program when uninstalled program number is reduced and when the domain keeping efficiency is given priority ~~to~~.

[0057] In addition, a ~~turn~~decision to uninstall may be decided by an installed date order, an importance, a number of a made document and number of a common file.

[0058] Fig. 13 and Fig. 14 are figures of sequence wherein the procedure of an application program being downloaded through the network 30 from the server 31 to the terminal 21, is shown.

[0059] In ~~the drawing~~Figure 13, the server 31 activates a management program, and terminal 21 activates a program for client. In addition, in HDD 27 of terminal 21, a history of a file acquired from server 31 by terminal 21 is stored as a file acquisition history file 1301.

[0060] At first, an installation process is started by installation control unit 91 of terminal 21, and a utilization start message is transmitted to the server 31 from the terminal 21. By this, the server 31 recognizes installation start, and the effectiveness of

an own automatic management system is notified to terminal 21.

[0061] Secondly, the terminal 21 notifies server 31 of a user identification mark (ID). When the server 31 ~~received~~receives this ID, the server 31 demands hardware information from terminal 21.

[0062] This hardware information is classification of CPU ~~(CPU)~~ 23 of terminal 21, clock frequency (Clock), the OS, memory capacity, and the volume of HDD 27. The server 31 demands a file acquisition history for terminal 21 after having received ~~these~~this information. This file acquisition history is so-called "history". This ~~is~~includes information such as a file name, URL (Uniform Resource Locator), compression system, a version, and the terminal 21 has ~~these~~this information as the file acquisition history file 1301 in HDD 27.

[0063] When the server 31 ~~received~~receives this file acquisition history, the following matters are judged based on this history. Whether ~~it~~the history was downloaded to the server in the past, and whether a new version of a file is already stored in a downloadable state in the server. ~~And when~~When a new version does not exist, the process finishes after an application program of a latest version ~~viewd~~is viewed from the server 31, is installed in the terminal 21.

[0064] On the other hand, when an application of a newer version exists in the server 31 than an application in a file acquisition history provided from the terminal 21, a necessary installer (an auxiliary program to execute installation) is selected, this is forwarded to terminal 21.

[0065] After ~~it is informed that~~the transfer of the installers from terminal 21 to server

31 is finished, the server 31 orders terminal 21 to renew a file acquisition history. Based on this order, terminal 21 demands a necessary item such as a version and file date from server 31. The server 31 replies to this and notifies the terminal 21 of information of the item.

[0066] Based on the item information, the terminal 21 renews own file acquisition history file 1301 and cuts off a network connected to the server 31. ~~And the~~ and terminal 21 activates a downloaded installer and starts installation. In this case, a legitimacy test of an archive begins when the installation is not finished normally. This archive legitimacy test is performed in the following ~~procedures~~process. Network connection with server 31 is performed again. A demand of test is transmitted to server 31. Based on this test result, when necessary, the installer is forwarded to the terminal 21 again by ~~the~~ server 31.

[0067] According to ~~this~~the present invention, an application program of the most suitable version can be decided for the data file, which is not specified from the appearance of a file, and the application program is installed automatically, thus automatic activation of the application becomes possible.

ABSTRACT OF THE DISCLOSURE

Without letting a user execute a cumbersome judgment of a version, an application program of the most suitable version is installed for data file automatically, thus an automatic activation of the application becomes possible. Firstly, a document file of a readout target is analyzed, a feature point as reserved word of macro instruction is extracted, and a version of an application suited for the document file is distinguished by an extracted feature point. Secondly, it is judged whether an application program of the version distinguished mentioned above is installed already. When found not being installed based on this judgment result, an application program of the version is newly installed.